



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION of  
Inventor(s): LIM

Appln. No.: 09 | 879,793  
Series Code ↑ | ↑ Serial No.

Group Art Unit: Unassigned

Filed: June 13, 2001

Examiner: Unassigned

Title: KEY SCHEDULER FOR ENCRYPTION APPARATUS  
USING DATA ENCRYPTION STANDARD ALGORITHM

Atty. Dkt. P 281441	P00HA010/US
M#	Client Ref

Date: October 4, 2001

**SUBMISSION OF PRIORITY  
DOCUMENT IN ACCORDANCE  
WITH THE REQUIREMENTS OF RULE 55**

Hon. Asst Commissioner of Patents  
Washington, D.C. 20231

Sir:

Please accept the enclosed certified copy(ies) of the respective foreign application(s) listed below for which benefit under 35 U.S.C. 119/365 has been previously claimed in the subject application and if not is hereby claimed.

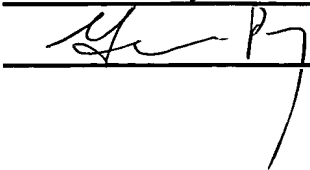
<u>Application No.</u>	<u>Country of Origin</u>	<u>Filed</u>
2000-32451	Korea	June 13, 2000

Respectfully submitted,

Pillsbury Winthrop LLP  
Intellectual Property Group

1600 Tysons Boulevard  
McLean, VA 22102  
Tel: (703) 905-2000

Atty/Sec: gjp/dlh

By Atty:	<u>Glenn J. Perry</u>	Reg. No.	<u>28458</u>
Sig:		Fax:	(703) 905-2500
		Tel:	(703) 905-2161



<Priority Document Translation>

THE KOREAN INDUSTRIAL  
PROPERTY OFFICE

This is to certify that annexed hereto is a true copy from the records of the Korean Industrial Property Office of the following application as filed.

Application Number : 2000-32451 (Patent)

Date of Application : June 13, 2000

Applicant(s) : HYUNDAI ELECTRONICS INDUSTRIES CO., LTD.

October 30, 2000

COMMISSIONER

Best Available Copy

Best Available Copy

HA/10

대한민국 특허청  
KOREAN INDUSTRIAL  
PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto  
is a true copy from the records of the Korean Industrial  
Property Office.

출원 번호 : 특허출원 2000년 제 32451 호  
Application Number

출원 년 월 일 : 2000년 06월 13일  
Date of Application

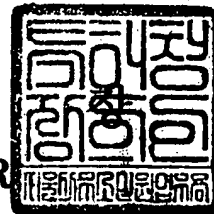
출원인 : 현대전자산업주식회사  
Applicant(s)



2000 년 10 월 30 일

특 허 청

COMMISSIONER



CERTIFIED COPY OF  
PRIORITY DOCUMENT

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2000.06.13
【발명의 명칭】	데이터 암호화 표준 알고리즘을 이용한 암호화 장치의 키 스케줄러
【발명의 영문명칭】	Key Scheduler of encryption device using data encryption standard algorithm
【출원인】	
【명칭】	현대전자산업주식회사
【출원인코드】	1-1998-004569-8
【대리인】	
【성명】	박해천
【대리인코드】	9-1998-000223-4
【포괄위임등록번호】	1999-008448-1
【대리인】	
【성명】	원석희
【대리인코드】	9-1998-000444-1
【포괄위임등록번호】	1999-008444-1
【발명자】	
【성명의 국문표기】	임영원
【성명의 영문표기】	LIM, Young Won
【주민등록번호】	621128-1067119
【우편번호】	467-850
【주소】	경기도 이천시 대월면 사동리 현대전자아파트 106-1302
【국적】	KR
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사를 청구합니다. 대리인 박해천 (인) 대리인 원석희 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	24 면 24,000 원

1020000032451

2000/11/

【우선권 주장료】	0	건	0	원
【심사청구료】	8	항	365,000	원
【합계】	418,000			원
【첨부서류】	1.	요약서·명세서(도면)_1통		

**【요약서】****【요약】**

본 발명은 8라운드로 작동하는 데이터 암호화 표준 알고리즘을 사용한 암호화 장치에서 보조키를 발생하는 키스케줄러에 관한 것으로, 레지스터의 면적을 반으로 줄인 암호화 장치의 키 스케줄러를 제공하는데 그 목적이 있다. 이를 위하여 본 발명의 키 스케줄러는 데이터 암호화 표준 알고리즘을 이용하여 암호화 연산을 수행하는 암호화 장치에 있어서, 56비트의 키를 입력받아 치환하는 제1치환선택부; 상기 제1치환선택부에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭에 의해서 저장하는 제1레지스터 및 제2레지스터; 상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 각각 왼쪽으로 미리 정한 비트수만큼 쉬프트하는 제1쉬프트부 및 제2쉬프트부; 상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 입력받아 48비트의 보조키를 생성하는 제2치환선택부; 상기 제1 및 제2레지스터의 출력을 입력받아서 미리 정한 비트수만큼 왼쪽으로 쉬프트하는 제3 및 제4쉬프트부; 및 상기 제3 및 제4쉬프트부의 출력을 입력받아서 치환하는 상기과 별개의 제2치환선택부를 포함하여 이루어진다.

**【대표도】**

도 8

**【색인어】**

제1치환선택부, 제2치환선택부, 제1쉬프트부, 제2쉬프트부, 제1레지스터, 제2레지스터.

**【명세서】****【발명의 명칭】**

데이터 암호화 표준 알고리즘을 이용한 암호화 장치의 키 스케줄러{Key Scheduller of encryption device using data encryption standard algorithm}

**【도면의 간단한 설명】**

도1은 DES 알고리즘을 설명하기 위한 블록도,

도2는 보조키(Subkey)를 발생하는 키 스케줄(Key Schedule)을 설명하기 위한 블록도,

도3은 일반적인 DES 코어의 아키텍처와 S-Box 치환부의 상세한 블록도,

도4는 루우프 전개 사이퍼 함수를 사용한 DES 코어의 아키텍처,

도5a와 도5b는 루우프 전개 사이퍼 함수에서 사용되는 제1 및 제2조합논리회로와 DES 아키텍처의 상세한 블록도,

도6은 루우프 전개 사이퍼 함수의 상세한 구성을 나타내는 블록도,

도7은 종래기술의 두 개의 연산부로 구성된 키 스케줄러 아키텍처를 설명하기 위한 블록도,

도8은 본 발명의 기본 연산부 하나만을 사용한 키 스케줄러 아키텍처의 구성을 나타내는 블록도,

도9는 기본 연산부만을 사용한 키 스케줄러의 또 다른 실시예를 도시한 블록도,

도10은 본 발명의 키 스케줄러 아키텍처의 동작을 나타내는 타이밍도,

도11은 본 발명의 시분할 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 블록도,

도12는 본 발명의 루우프 전개 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 구성을 나타내는 블록도,

도13은 본 발명의 상기 루우프 전개 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 동작 순서를 나타내는 타이밍도,

도14는 매크로 파이프라인의 효과를 나타낸 타이밍도.

**\* 도면의 주요부분에 대한 부호의 설명 \***

800 : 제1치환선택부

810 : 제1레지스터

820 : 제2레지스터

830 : 제1쉬프트부

840 : 제2쉬프트부

850 : 제2치환선택부

**【발명의 상세한 설명】**

**【발명의 목적】**

**【발명이 속하는 기술분야 및 그 분야의 종래기술】**

<19> 본 발명은 데이터 암호화 표준 알고리즘을 이용한 암호화 장치에 관한 것으로, 특히 키를 발생하는 키 스케줄러에 관한 것이다.

<20> 일반적으로 데이터 암호화 표준(DES : Data Encryption Standard, 이하 DES



라 칭함) 알고리즘은 가장 널리 쓰이고 있는 암호화 방식으로 네트워킹 사용이 증가함에 따라 그 중요성을 더해 가고 있다. 특히, 보안 인터넷 응용이나 원격 접근 서버나 케이블 모뎀과 위성용 모뎀 등의 분야에서 많이 이용되고 있다.

<21> DES는 기본적으로 64비트 블록의 입력 및 출력을 가지는 64비트 블록 암호이며, 64비트의 키 블록 중 56비트가 암호화 및 복호화에 사용되고, 나머지 8비트는 패리티 검사용으로 사용된다. 또한, 64비트의 평문(Plain Text) 블록과 56비트의 키(Key)를 입력으로 해서 64비트의 암호문(Cipher Text) 블록을 출력하는 암호화 장치이다.

<22> 도1은 DES 알고리즘을 설명하기 위한 블록도이다.

<23> 상기 도1을 참조하면, 종래기술의 암호화 장치의 알고리즘은 먼저 64비트의 평문(Plain Text) 블록을 초기 치환(IP : Initial Permutation)에 의해 치환을 수행하는 초기치환부(110)와, 다음에 상기 치환부(110)의 64비트의 블록을 두개의 32비트 블록으로 나누어 왼쪽변수( $L_i$ )와 오른쪽변수( $R_i$ )에 저장한 후, 사이퍼(Cipher) 함수  $f$ 로 수행되는 16라운드의 곱 변형(Product Transformation)과 왼쪽변수( $L_i$ )와 오른쪽변수( $R_i$ )를 매라운드마다 교환하는 16라운드의 블록 변형(Block Transformation)을 수행하는 기본연산부(120)와, 16라운드에 걸친 변형이 끝난 후 역초기 치환( $IP^{-1}$ )을 거쳐서 암호화된 암호문(Cipher Text)을 출력하는 역초기치환부(130)로 구성되어 있다.

<24> 상기 기본연산부(120)에서의 곱 변형은 오른쪽변수( $R_i$ )에 저장된 32비트의 블록을 키 스케줄(Key Schedule)에 의해서 생성된 보조키(Subkey)  $K_i$ 와 함께 입력시켜서 암호화 연산을 수행하는 사이퍼 함수부( $f$ )(121)와, 상기 사이퍼 함수  $f$ 의 결과를 왼쪽변수( $L_i$ )에 저장된 32비트의 블록과 함께 배타적 논리합하는 익스쿠르시브-오아부(122)로 구성되어 있다.

<25> 상기 익스курс티브-오아부(122)의 32비트의 데이터는 다음 라운드의

오른쪽변수( $R_{i+1}$ )에 저장되고 상기 오른쪽변수( $R_i$ )에 저장된 32비트의 데이터는 다음 라운드의 왼쪽변수( $L_{i+1}$ )에 교환(Swapping)되어 저장됨으로 블록 변형이 수행된다. 이러한 1라운드의 연산이 반복되어 16라운드가 수행되는 것이다.

<26> 초기치환부(110)을 거친 64비트의 평문(Plain Text) 블록을 둘로 나누어 왼쪽변수( $L_0$ )와 오른쪽변수( $R_0$ )에 입력하면 16회의 각 라운드는 다음 수학식1과 수학식2로 표현 가능하다.

<27> 【수학식 1】

$$L_i = R_{i-1} \quad i=1,2,\dots,16$$

<28> 【수학식 2】

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad i=1,2,\dots,16$$

<29> 도2는 보조키(Subkey)를 발생하는 키 스케줄(Key Schedule)을 설명하기 위한 블록도이다.

<30> 상기 도2를 참조하면, 키 스케줄은 56비트의 키(Key)를 입력받아서 제1치환선택부(PC1 : Permutation Choice 1)(200)에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 변수  $C_0$ 와  $D_0$ 에 저장한 후, 제1 및 제2쉬프트부(220, 230)에 의해 변수  $C_i$ 와  $D_i$ 에 ( $i=1, 2, \dots, 16$ ) 저장된 값을 각각 왼쪽으로 한자리 또는 두자리씩 쉬프트하며 다음 라운드의 변수  $C_{i+1}$ 과  $D_{i+1}$ 에 저장함과 동시에, 상기 변수  $C_i$ 와  $D_i$ 에 ( $i=1, 2, \dots, 16$ ) 저장된 두개의 28비트의 블록을 입력받는 제2치환선택부(PC2 : Permutation Choice 2)(240)에 의해 치환된 48비트의 보조키(Subkey)  $K_i$ 를 출력함으로써 16라운드의

연산에 필요한 보조키를 생성한다.

- <31> 16라운드 동안  $C_i$ 와  $D_i$ 는 28자리 수만큼 쉬프트가 되어  $C_0$ 와  $C_{16}$  그리고  $D_0$ 와  $D_{16}$ 은 서로 같은 데이터가 된다.
- <32> 도3은 일반적인 DES 코아의 아키텍처와 S-Box 치환부의 상세한 블록도이다.
- <33> 상기 도3을 참조하면, DES 코아는 32비트의 텍스트 블록  $R(i-1)$ 을 저장하고 있는 오른쪽레지스터로부터 32비트의 데이터를 입력받아 48비트의 데이터로 확장 치환하는 확장치환부(310)와, 상기 확장치환부의 48비트의 데이터를 입력받고 키 스케줄(Key Schedule)로부터의 보조키( $K_i$ )를 입력받아 배타적 논리합 연산을 수행하는 익스쿠르시브-오아부(320)와, 상기 익스쿠르시브-오아부(320)로부터의 48비트의 데이터를 32비트의 데이터로 대치 치환하는 S-Box 치환부(330)와, 상기 S-Box 치환부(330)의 32비트의 데이터를 복사 치환하는 P-Box 치환부(340)와, 상기 P-Box 치환부의 32비트의 데이터와 왼쪽레지스터에 저장되어 있는 32비트의 블록  $L(i-1)$ 을 입력받아 배타적 논리합하는 익스쿠르시브-오아부(350)를 구비한다.
- <34> 키 스케줄(Key Schedule)은 제1치환선택부(PC1 : Permutation Choice 1)(360)으로부터 56비트의 키(Key)를 입력받아서 28비트의 두 블록으로 나누어서 각각 왼쪽으로 한 자리 또는 두자리씩 쉬프트하는 쉬프트부(370, 380)와, 상기 28비트의 두 블록을 입력시켜 48비트의 보조키를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(390)를 구비한다.
- <35> 구체적으로, 상기 S-Box 치환부는 48비트의 입력을 받아서 32비트의 출력을 생성하는 8 개의 S-Box로 구성되어 있다. 즉, 48비트의 데이터는 8 개의 6비트 데이터로 분할

되어 8 개의 S-Box에 입력된다. 이 8 개의 S-Box들은 8 개의 4비트 출력을 내보냄으로써 48 비트를 32비트로 줄인다. S-Box 치환부(330)는 테이블 룩-업(Look-up) 방식으로 대치됨으로써 프로그램가능 논리 어레이(PLA)나 롬(ROM)과 같은 기억장치를 필요로 한다. 6비트의 입력에 대하여 4비트를 출력하기 때문에 각 S-Box는  $64 \times 4$  의 기억 용량이 필요하며 전체적으로 8개의 S-Box로 구성되어 있으므로  $8 \times 64 \times 4$ 의 기억장치가 필요하다. 따라서 전체적으로 칩에서 차지하는 면적이 상대적으로 크다

<36> 이상과 같이 동일한 연산 구조를 16회 반복하는 루우프로 구성된 DES 알고리즘을 구현하기 위해서, 종래의 아키텍처는 도3과 같이 한 개의 라운드를 기본 연산부로 하여 하드웨어를 구현하였으나, 최근 2 내지 16개의 라운드를 기본 연산부로 하는 루우프 전개 아키텍처가 대두되고 있다. 이 방식은 합성시 각 라운드간의 슬랙(Slack, Time Margin)을 효율적으로 줄이고 주변 최적화(Boundary Optimization) 합성 기법을 통해 집적화시 면적을 줄이는 효과가 있다. 이 루우프 전개 아키텍처는 한 클럭 사이클 당 두 개의 라운드를 연산하기 때문에, 암호화를 8 클럭 사이클 안에 수행할 수 있으나, 집적화시에 면적을 크게 차지하는 S-Box 치환부가 두개 필요하다.

<37> 도4는 루우프 전개 사이퍼 함수를 사용한 DES 코어의 아키텍처이다.

<38> 상기 도4를 참조하면, 루우프 전개 사이퍼 함수를 사용한 DES코어의 아키텍처는 데이터와 키(Key)를 입력받아 치환하는 초기 치환부(400)와, 상기 초기 치환부(400)에서 치환된 데이터와 제1레지스터(450)를 통하여 피드백된 데이터 중 하나를 선택하는 제1멀티플렉서(410)와, 상기 초기치환부에서 치환된 키와 제2레지스터(460)를 통하여 피드백된 키 중 하나를 선택하는 제2멀티플렉서(420)와, 상기 제1 및 제2멀티플렉서로부터 출력된 데이터와 키를 입력받아 홀수번째 라운드의 암호화 연산을 수행하는 제1조합논리회

로부(430)와, 상기 제1조합논리회로부(430)의 출력을 입력받아 짝수번째 라운드의 암호화 연산을 수행하는 제2조합논리회로부(440)와 상기 제2조합논리회로부(440)로부터 출력된 데이터를 저장하는 제1레지스터(450)와, 상기 제2조합논리회로부(440)로부터 출력된 키를 저장하는 제2레지스터(460)와, 상기 제1레지스터(450)로부터 출력된 데이터를 입력받아 치환하여 최종 암호문을 생성하는 최종 치환부(470)를 구비한다.

<39> 도5a와 도5b는 루우프 전개 사이퍼 함수에서 사용되는 제1 및 제2조합논리회로와 DES 아키텍처의 상세한 블록도이다.

<40> 상기 도5a와 도5b를 참조하면, 도4의 제1 및 제2조합논리회로는 DES 알고리즘의 두 개 라운드를 하드웨어로 구현한 것으로 도5a에 도시되어 있는 것과 같이 루우프 전개 사이퍼 함수와 키 스케줄러의 조합 논리회로 부분을 나타낸다. 한 클럭 사이클 동안 키 스케줄러가 두개의 보조키  $K_m$  및  $K_n$ 을 생성하면, 루우프 전개 사이퍼 함수는 이들의 보조키를 입력받아 두 개의 사이퍼 함수  $f_m$  및  $f_n$ 과 두개의 배타 논리합 연산자들로 구성된 DES 알고리즘의 두개 라운드를 수행한다. 즉 루우프 전개 사이퍼 함수는 두개의 보조키  $K_m$  및  $K_n$ 과 레지스터 A,B의 출력  $R_A$  및  $R_B$ 를 입력으로 받아 두 개의 연산된 출력  $R_C$ 와  $R_D$ 를 다음 클럭 사이클에서 해당 레지스터에 입력한다.

<41> 도6은 루우프 전개 사이퍼 함수의 상세한 구성을 나타내는 블록도이다.

<42> 상기 도6을 참조하면, 루우프 전개 사이퍼 함수는 32비트의 텍스트 블록을 저장하고 있는 레지스터( $R_B$ )로부터 32비트의 데이터를 입력받아 48비트의 데이터로 확장 치환하는 확장치환부(610)와, 상기 확장치환부(610)의 48비트의 데이터를 입력받고 키 스케줄(Key Schedule)로부터의 보조키( $K_m$ )를 입력받아 배타적 논리합 연산을 수행하는 익스쿠르시브-오아부(620)와, 상기 익스쿠르시브-오아부(620)로부터의 48비트의 데이터를 32

비트의 데이터로 대치 치환하는 S-Box 치환부(630)와, 상기 S-Box 치환부(630)의 32비트의 데이터를 복사 치환하는 P-Box 치환부(640)와, 상기 P-Box 치환부(640)의 32비트의 데이터와 레지스터( $R_A$ )에 저장되어 있는 32비트의 데이터를 입력받아 배타적 논리합하여 레지스터( $R_C$ )로 출력하는 익스쿠르시브-오아부(650)를 구비하며, 이러한 구성을 갖고 레지스터( $R_D$ )로 데이터를 출력하는 또 다른 루우프 전개 사이퍼 함수를 하나 더 구비한다.

<43> 레지스터 A, B에 32비트의 데이터  $R_{2i-3}$ ,  $R_{2i-2}$ 이 저장되어 있고 키 스케줄러가 연속된 두개의 키  $K_{2i-1}$ 과  $K_{2i}$ 를 공급한다고 가정하면, 다음 수학식3과 수학식4에 의해 32비트의 데이터  $R_{2i-1}$ 과  $R_{2i}$ 의 값들을 한 클럭 사이클 동안 연산한다.

<44> 【수학식 3】

$$R_{2i-1} = R_{2i-3} \oplus f(R_{2i-2}, K_{2i-1}) \quad i=1,2,\dots,8$$

<45> 【수학식 4】

$$R_{2i} = R_{2i-2} \oplus f(R_{2i-1}, K_{2i}) \quad i=1,2,\dots,8$$

<46> 도7은 종래기술의 두 개의 연산부로 구성된 키 스케줄러 아키텍처를 설명하기 위한 블록도이다.

<47> 상기 도7을 참조하면, 종래기술의 키 스케줄러는 56비트의 키(Key)를 입력받아서 치환하는 제1치환선택부(PC1 : Permutation Choice 1)(700)와, 상기 제1치환선택부(700)에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 제1클럭(CLK1)에 의해서 저장하는 제1레지스터( $C_m$ )(710) 및 제2레지스터( $D_m$ )(720)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블록을 각각 왼쪽으로 두자리, 세자리 또는 네자리씩 쉬프트하는 제1쉬프트부(730) 및 제2쉬프트부(740)와,

상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블록을 입력받아 48비트의 보조키( $K_m$ )를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(750)로 8라운드로 동작하는 제1유닛이 구성되고 제1유닛과 같은 구성을 갖는 제2유닛이 더 포함된다.

<48> 8라운드 동안 상기 제1유닛은  $K_{2i-1}$ 을, 제2유닛은  $K_{2i}$ 를 생성하게 한다. 즉 제1유닛은 누적된 쉬프트 비트 수가 1, 4, 8, 12, 15, 19, 23, 27이 되도록 8 클럭 사이클 동안 1비트, 2비트, 3비트, 또는 4비트씩 왼쪽으로 쉬프트시키고, 제2유닛은 누적된 비트 수가 2, 6, 10, 14, 17, 21, 25, 28이 되도록 8 클럭 사이클 동안 2비트, 3비트, 또는 4비트씩 왼쪽으로 쉬프트시킨다.

<49> 상기과 같은 구성을 가지고 한 클럭당 두 개의 보조키를 발생시키는 키 스케줄러는 도2에서 도시한 바와 같이 두 개의 28비트 레지스터와 왼쪽 쉬프트를 사용하여 각 라운드마다 1비트 또는 2비트씩 왼쪽으로 쉬프트하여 한 개의 보조키를 발생시키는 키스케줄러에 비교해서 모두 네 개의 레지스터와 쉬프트가 필요하며 레지스터가 차지하는 면적이 커서 집적화시 문제가 된다.

#### 【발명이 이루고자 하는 기술적 과제】

<50> 본 발명은 상기과 같은 종래 기술의 문제점을 해결하기 위하여 안출된 것으로서, 레지스터의 면적을 반으로 줄인 암호화 장치의 키 스케줄러를 제공하는데 그 목적이 있다.

#### 【발명의 구성 및 작용】

<51> 상기 목적을 달성하기 위하여 본 발명의 키 스케줄러는 데이터 암호화 표준 알고리

즘을 이용하여 암호화 연산을 수행하는 암호화 장치에 있어서, 56비트의 키를 입력받아 치환하는 제1치환선택부; 상기 제1치환선택부에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭에 의해서 저장하는 제1레지스터 및 제2레지스터; 상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 각각 왼쪽으로 미리 정한 비트수만큼 쉬프트하는 제1쉬프트부 및 제2쉬프트부; 상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 입력받아 48비트의 보조키를 생성하는 제2치환선택부; 상기 제1 및 제2레지스터의 출력을 입력받아서 왼쪽으로 미리 정한 비트수만큼 쉬프트하는 제3 및 제4쉬프트부; 및 상기 제3 및 제4쉬프트부의 출력을 입력받아서 치환하는 상기과 별개의 제2치환선택부를 포함하여 이루어진다.

<52> 이하, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 정도로 상세히 설명하기 위하여, 본 발명의 가장 바람직한 실시예를 첨부한 도면을 참조하여 설명하기로 한다.

<53> 도8은 본 발명의 기본 연산부 하나만을 사용한 키 스케줄러 아키텍처의 구성을 나타내는 블록도이다.

<54> 상기 도8을 참조하면, 본 발명의 키 스케줄러는 56비트의 키(Key)를 입력받아 치환하는 제1치환선택부(800)와, 상기 제1치환선택부(800)에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭(CLK)에 의해서 저장하는 제1레지스터( $C_m$ )(810) 및 제2레지스터( $D_m$ )(820)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블록을 각각 왼쪽으로 두자리, 세자리 또는 네자리씩 쉬프트하는 제1쉬프트부(830) 및 제2쉬프트부(840)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블록을 입력받아 48



비트의 보조키( $K_m$ )를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(850)와, 상기 제1 및 제2레지스터(810, 820)의 출력을 입력받아서 왼쪽으로 한자리 또는 두자리씩 쉬프트하는 제3 및 제4쉬프트부(860, 870)와, 상기 제3 및 제4쉬프트부(860, 870)의 출력을 입력받아서 48비트의 보조키를 생성하는 제2치환선택부(PC2)(880)를 구비한다.

<55>      상기 본 발명의 키 스케줄러는 8라운드로 동작하는 기본 연산부 제1유닛 하나만을 사용해서  $i$ 번째 라운드에서 보조키  $K_{2i-1}$ 을 연산한다. 상기 제1 및 제2레지스터( $C_m$ ,  $D_m$ )은 상기 제1치환선택부(800)를 거친 초기 키를 입력받거나, 상기 제1 및 제2레지스터(810, 820)의 출력을 제1 및 제2쉬프트부(830, 840)에 의해 쉬프트 시킨 값을 다음 클럭 사이클에서 저장한다. 각 라운드에서 왼쪽 쉬프터가 쉬프트하는 비트수  $S_m$ 은 3, 4, 4, 3, 4, 4, 4, 2(1)이다.

<56>      도8의 도표에서 알 수 있듯이,  $i$ 번째 라운드에서 보조키  $K_{2i-1}$ 을 얻기위한 누적된 쉬프트 비트 수  $TS_m$ 과 보조키  $K_{2i}$ 를 얻기위한 누적된 쉬프트 비트수  $TS_n$ 은,  $TS_n - TS_m = D_m$ 이라 할 때 다음과 같은 관계가 있다. 첫번째 라운드( $P_0$ )에서 변수  $D_m=1$ 이고, 여덟번째 라운드( $P_7$ )에서 초기키를 새로이 저장할 경우에는 변수  $D_m=0$ 이며, 동일한 키를 사용하여 반복해서 비암호화 입력 데이터를 암호화할 경우에는 변수  $D_m=1$ 이다. 나머지 라운드( $P_1$  내지  $P_6$ )에서는 변수  $D_m=2$ 이다. 따라서, 도8과 같이 왼쪽 쉬프터(860, 870)와 와이어링으로 구현되는 제2치환선택부(PC2)(880)를 추가로 사용하여,  $i$ 번째 라운드에서  $K_{2i-1}$ 값과  $K_{2i}$ 값을 연산하여  $K_m$ 과  $K_n$ 으로 각각 출력할 수 있다.

<57>      도9는 기본 연산부만을 사용한 키 스케줄러의 또 다른 실시예를 도시한 블록도이다

<58>      상기 도9를 참조하면, 키 스케줄러는 56비트의 키를 입력받는 제1치환선택부 (PC1)(900)와, 상기 제1치환선택부(900)의 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭(CLK)에 의해서 저장하는 제1레지스터( $C_n$ )(910) 및 제2레지스터 ( $D_n$ )(920)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블록을 각각 왼쪽으 로 두자리, 세자리 또는 네자리씩 쉬프트하는 제1쉬프트부(930) 및 제2쉬프트부(940)와, 상기 제1레지스터 및 제2레지스터의 28비트의 블록을 입력받아 48비트의 보조키( $K_n$ )를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(950)와, 상기 제1 및 제2레지스터 (910, 920)의 출력을 입력받아서 오른쪽으로 한자리 또는 두자리씩 쉬프트하는 제3 및 제4쉬프트부(960, 970)와, 상기 제3 및 제4쉬프트부(960, 970)의 출력을 입력받아서 48 비트의 보조키( $K_m$ )을 생성하는 제2치환선택부(PC2)(980)을 구비한다.

<59>      상기 도9에서 도시된 키 스케줄러는 도7의 기본 연산부 제2유닛만을 사용하여  $i$ 번 째 라운드에서 보조키  $K_{2i}$ 를 연산한다. 도9의 도표에서 알 수 있듯이,  $i$ 번째 라운드에서 보조키  $K_{2i-1}$ 을 얻기위한 누적된 쉬프트 비트수  $TS_m$ 과 보조키  $K_{2i}$ 를 얻기 위한 누적된 쉬프트 비트수  $TS_n$ 은  $TS_m - TS_n = D_n$ 이라고 할 때 다음과 같은 관계가 있다. 첫번째 라운드( $P_0$ )와 여덟번째 라운드( $P_7$ )에서  $D_n = -1$  이고, 나머지 라운드( $P_1$  내지  $P_6$ )에서는  $D_n = -2$ 이다. 따라서, 도9와 같이 오른쪽 쉬프트부(960, 970)와 와이어링으로 구현되는 제2치 환선택부(PC2)(980)를 추가로 사용하여,  $i$ 번째 라운드에서  $K_{2i}$  값과  $K_{2i-1}$ 값을 연산하여  $K_n$ 과  $K_m$ 으로 각각 출력할 수 있다.

<60>      도10은 본 발명의 키 스케줄러 아키텍처의 동작을 나타내는 타이밍도이다.

<61>      상기 도 10을 참조하면,  $K_m$ 과  $K_n$ 는 8라운드 DES 아키텍처에서 16개의 보조키가 엑

세스되는 시간 구간을 나타낸다.  $TS_m$  및  $TS_n$ 는 필요한 보조키를 얻기 위해 PC1을 거친 초기 키가 왼쪽 쉬프트에 의해 쉬프트된 총 비트 수를 나타낸다.  $S_m$  및  $S_n$ 는  $TS_m$  및  $TS_n$ 에 표시된 총 쉬프트된 비트 수를 얻기 위해서 각 라운드 ( $P_i$ )에서 쉬프트하는 비트 수를 나타낸다. 본 발명에서 보조키를 생성하는 과정은 다음과 같다.

- <62> 먼저, 제 1 라운드 ( $P_0$ )에서  $TS_m$  및  $TS_n$ 는 1과 2로서 해당 레지스터의 출력, 즉 PC1을 거친 초기키를 1비트와 2비트씩 왼쪽 쉬프트된 결과를 PC2로 출력하여 보조키  $K_1$ 과  $K_2$ 를 생성할 수 있다.
- <63> 제 2 라운드 ( $P_1$ )에서 보조키  $K_3$ 와  $K_4$ 를 생성하기 위해서, 즉  $TS_m=4$ 와  $TS_n=6$ 을 위해서 제 1 라운드에서 왼쪽 쉬프트는 해당 레지스터를  $3(=4-1)$ 비트와  $4(=6-2)$ 비트 씩 왼쪽 쉬프트한다.
- <64> 제3라운드 ( $P_2$ )에서 보조키  $K_5$ 와  $K_6$ 를 생성하기 위해서, 즉  $TS_m=8$ 와  $TS_n=10$ 을 위해서 제 2 라운드에서 왼쪽 쉬프트는 해당 레지스터를  $4(=8-4)$ 비트와  $4(=10-6)$ 비트 씩 왼쪽 쉬프트한다.
- <65> 이와 같이 각 라운드 ( $P_i$ )에서 해당 레지스터를  $S_m$  또는  $S_n$ 비트씩 왼쪽 쉬프트하여 제 8 라운드 ( $P_7$ )에서  $TS_A=27$ 와  $TS_B=28(=0)$ 비트만큼 초기키가 쉬프트된다. 다음에 다시 제 1 라운드로 돌아가기 위해서, 즉  $TS_m=1$ 과  $TS_n=2$ 가 되기 위해서,  $S_m=2$  및  $S_n=2$ 가 되어야 한다.
- <66> 일반적으로 주어진 키에 대하여 암호화 해야될 데이터가 다수인 경우가 많다. 이 때 파이프라인을 사용하여 암호화하는 성능을 높일 수 있다. DES 아키텍처에서 사용되는 파이프라인은 적용되는 레벨에 따라 마이크로 파이프라인(Micro Pipeline)과 매크로 파

이프라인(Macro Pipeline)의 두 종류로 구분할 수 있다.

<67> DES 코어에 입력되는 데이터의 속도는 DES 코어보다도 DES 코어 외부의 전체 시스템 측면에서 결정된다. 네트워크에 사용되는 DES의 경우 변조기 및 복조기의 최대 전송 속도 및 외부 호스트 마이크로 프로세서의 속도 등을 고려하여 정해진다. 일반적으로 DES에 입출력되는 속도는 느리다. 그리고 DES코어 외부의 시스템에서 데이터는 대부분 바이트(8비트) 단위로 이동되며 DES 코어는 64비트의 입력 데이터를 64비트로 암호화하여 출력하기 때문에, 8 개의 입력 바이트들을 모아서 DES 코어에 전달하는 입력 버퍼 레지스터(IBR)와 64비트의 DES 출력을 바이트 단위로 전달하기 위한 출력 버퍼 레지스터(OBR)를 사용하여 8 클럭 사이클 동안 데이터 포맷팅을 수행하여야 한다. 이와같이 느린 입출력 과정의 레이턴시를 숨기기 위하여 도11 및 도12와 같은 입력과정(제1단계), DES 연산과정(제 2단계), 출력 과정(제3단계)으로 3단계 파이프라인을 구현한 것을 매크로 파이프라인으로 구별하자. 매크로 파이프라인의 주기는 입력 및 출력 과정에 걸리는 시간과 DES 연산 시간 중 최대 값으로 결정되며, 이 들 시간이 동일할 때 최대의 효과를 얻을 수 있다.

<68> 도11은 시분할 사이퍼 함수를 이용한 DES 코어에 3단계 매크로 파이프라인을 적용한 아키텍처이고, 도12는 루우프 전개 사이퍼 함수를 이용한 DES 코어에 3단계 매크로 파이프라인을 적용한 아키텍처이다.

<69> 도11은 본 발명의 시분할 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 블록도이다.

<70> 상기 도11의 DES 아키텍처는 64비트의 입력 데이터를 8개로 분할하여 순차적으로 8 비트씩 입력받아 제1클럭(CLK1)에 따라서 왼쪽 입력 버퍼 레지스터(IBR(L))(1110)와 오

른쪽 입력 버퍼 레지스터(IBR(R))(1120)에 각각 32비트의 입력데이터를 수집하여 저장하는 제1단계와, 상기 왼쪽 입력 버퍼 레지스터와 상기 오른쪽 입력 버퍼 레지스터로부터 각각의 32비트의 데이터 블록을 제1사이퍼 함수와 제2사이퍼 함수에 교대로 입력시켜 8라운드의 암호화 연산을 수행하는 제2단계와, 상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터(OBR(L))(1140)과 오른쪽 출력 버퍼 레지스터(OBR(R))(1150)을 통하여 각각 8비트씩 분배하여 출력하는 제3단계로 이루어진다.

<71>      상기 도11에서의 시분할 사이퍼 함수의 입력은 레지스터 A0와 B0의 32비트 출력과 키 스케줄러의 48비트 출력  $K_A$ 와  $K_B$ 로 구성된다. 상기 제1클럭(CLK1) 주기의 전반부에서는 사이퍼 함수  $f_A$ 를 연산하여 출력하게 하며, 상기 제1클럭(CLK1) 주기의 후반부에서는 사이퍼 함수  $f_B$ 를 연산하여 출력하게 한다. 즉 시분할 사이퍼 함수는 상기 제1클럭(CLK1) 주기의 전반부에서 A0와  $K_A$ 를 입력해서 확장 치환부, 배타적 논리합, S-Box 치환부, 그리고 P-Box 치환부로 구성되는 사이퍼 함수를 연산하여  $f_A$ 로 출력하고, 마찬가지로 상기 제1클럭(CLK1) 주기의 후반부에서 B0와  $K_B$ 를 입력받아 사이퍼 함수 연산 결과를  $f_B$ 로 출력한다.

<72>      따라서, 한 클럭 사이클 동안 두 개의 연속된 보조키 쌍  $K_{2i-1}$ 과  $K_{2i}$ 를 생성하는 본 발명의 키 스케줄러 아키텍처들은 상기 도11의 시분할 사이퍼 함수를 이용하여 8 라운드로 동작하는 아키텍처에도 사용할 수 있다.

<73>      도12는 본 발명의 루우프 전개 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 구성을 나타내는 블록도이다.

<74>      상기 도12를 참조하면, DES 아키텍처는 64비트의 입력 데이터를 8개로 분할하여 순

차적으로 8비트씩 입력받아 제1클럭(CLK1)에 따라서 왼쪽 입력 버퍼 레지스터 (IBR(L))(1210)와 오른쪽 입력 버퍼 레지스터(IBR(R))(1220)에 각각 32비트의 입력데이터를 수집하여 저장하는 제1단계와, 상기 왼쪽 입력 버퍼 레지스터 및 오른쪽 입력 버퍼 레지스터로부터 두 개의 32비트 블록을 입력받아 8라운드에 거쳐서 루우프 전개 사이퍼 함수 연산을 수행하여 암호화하는 제2단계와, 상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터(OBR(L))(1260)과 오른쪽 출력 버퍼 레지스터(OBR(R))(1270)을 통하여 각각 8비트씩 분배하여 출력하는 제3단계로 이루어진다.

<75> 한 클럭 사이클 동안 두 개의 연속된 보조키 쌍  $K_{2i-1}$ 과  $K_{2i}$  ( $i=1,2,\dots,8$ )를 생성하는 본 발명의 키 스케줄러 아키텍처들은 상기 루우프 전개 사이퍼 함수를 사용한 DES 아키텍처와 상기 시분할 사이퍼 함수를 사용한 DES 아키텍처에 적용 가능하다. 종래기술의 키 스케줄러 아키텍처는 28비트의 쉬프트 된 결과를 저장하는 4 개의 레지스터를 사용하여, 집적화 시 레지스터가 차지하는 면적이 커진다.  $i$ 번째 라운드에서 보조키  $K_{2i-1}$  또는  $K_{2i}$ 를 얻기 위한 누적된 쉬프트 비트수는 1비트 또는 2비트의 차이가 있다. 그러므로 키 스케줄러 아키텍처의 기본 연산부(도7의 제1유닛이나 제2유닛)만을 이용하여 도8과 도9의 키 스케줄러 아키텍처를 구현할 수 있다. 이 아키텍처들은 보조키  $K_{2i-1}$  또는  $K_{2i}$ 를 연산하는데 사용되는 레지스터에 저장된 값으로부터 왼쪽 쉬프트터 또는 오른쪽 쉬프트터와 제2치환선택부(PC2)를 추가로 사용하여 다른 보조키  $K_{2i}$ 와  $K_{2i-1}$ 을 각각 구할 수 있다. 기본연산부 제1유닛만을 사용한 도8의 아키텍처는 누적된 비트수가 0(=28)이 되는 것이 없기 때문에 초기키를 저장할 경우 별도로 처리하여야 한다. 그러므로 기본 연산부 제2유닛만을 사용한 도9의 아키텍처에 비해서 사용되는 조합 논리회로로 구현된 쉬프트터의 크기가 커진다. 루우프 전개 사이퍼 함수는 먼저  $K_{2i-1}$ 을 사용하여 사이퍼 함수 연산

을 시작하지만, 기본 연산부 제2유닛만을 사용한 도9의 아키텍처는 조합 논리회로로 구현된 오른쪽 쉬프터를 통해서  $K_{2i-1}$ 을 얻기 때문에 기본 연산부 제1유닛만을 사용한 도8의 아키텍처에 비해서 임계 경로를 증가시킨다.

<76> 도13은 본 발명의 상기 루우프 전개 사이퍼 함수를 이용한 DES 코어에 매크로 파이프라인을 적용한 DES 아키텍처의 동작 순서를 나타내는 타이밍도이다.

<77> 상기 도13를 참조하면, 본 발명의 DES 아키텍처가 비암호화 입력 데이터  $(y_0, z_0)$ ,  $(a_0, b_0)$ ,  $(c_0, d_0)$ 를 순서대로 입력받아  $z_i, b_i, d_i, (i=1,2,\dots,16)$ 를 계산하여  $(z_{16}, z_{15}), (b_{16}, b_{15}), (d_{16}, d_{15})$ 을 차례로 출력하는 과정 중 비암호화 데이터  $a_0$ 와  $b_0$ 로부터  $b_1, b_2, \dots, b_{16}$ 을 계산하여  $b_{16}$ 과  $b_{15}$ 를 출력하는 과정을 중점으로 나타내었다.  $a_0$ 와  $b_0$ 를 초기 치환 IP를 거친 64비트의 비암호화 블록이 32비트의 두블럭으로 나뉘어진 것을 나타낸다고 하자. 즉  $a_0=L_0=R_{-1}$ 이고  $b_0=R_0$ 이다. 그리고 DES 코어가 계산한 값을  $b_1, b_2, \dots, b_{16}(b_i=R_i)$ 라고 한다.  $b_i$ 를 계산하기 전에 미리 키 스케줄러가 적절한 보조키  $K_i$ 를 사이퍼 함수에 입력해주도록 하면, 본 발명의 아키텍처에서 DES가 연산되는 과정은 다음과 같다.

<78> 먼저, 입력 과정은 시간  $t_0$ 이전에 8 사이클 동안 바이트 단위로 입력된 데이터를 레지스터 IBR에 수집한다. 그리고 시간  $[t_0-t_2]$  구간에서 레지스터 IBR(L)은  $a_0$ 를, 레지스터 IBR(R)은  $b_0$ 를 유지한다. 동시에 다음 비암호화 입력 데이터  $c_0$ 와  $d_0$ 를 한 바이트씩 레지스터 IBR에 수집한다. 따라서 8 사이클 후인 시간  $[t_{16}-t_{18}]$ 에서 레지스터 IBR은  $c_0$ 와  $d_0$ 의 값을 유지한다.

<79> 출력 과정을 살펴보면, 레지스터 OBR은 시간  $t_1$ 에서 레지스터 A와 B에서  $z_{16}$ 과  $z_{15}$

의 값을 로드한 후 역치환된 데이터를 시간  $t_2$ 에서 한 바이트씩 8 사이클 동안 출력한다.  $z_{16}$ 과  $z_{15}$ 의 값은 레지스터 OBR에서 시간  $[t_1-t_{17}]$ 구간에서 유지되며, 시간  $t_{17}$ 에 다시 레지스터 A와 B의 값( $b_{16}$ 과  $b_{15}$ )을 로드한 후 8 사이클 동안 유지하여 시간  $t_{17}$ 에서부터 다시 역치환된 출력 데이터를 한 바이트씩 8 사이클 동안 출력한다.

<80> DES 연산과정은 시간  $[t_0-t_2]$  구간에서 레지스터 IBR(R)과 IBR(L)에 저장된 값  $a_0$ 와  $b_0$ 를 액세스하고 시간  $[t_0-t_2]$ 구간에서 보조키  $K_1$ 과  $K_2$ 를 키 스케줄러로부터 입력받으면 시간  $[t_0-t_2]$ 구간에서 루우프 전개 사이퍼 함수를 연산하여 시간  $t_2$ 에서 연산된  $b_1$ 과  $b_2$ 의 값을 레지스터 A와 B에 저장한다.

<81> 또한, 시간  $[t_2-t_4]$  구간에서 레지스터 A와 B로부터  $b_1$ 과  $b_2$  값을 액세스할 수 있으므로, 시간  $[t_2-t_4]$ 구간에서 보조키  $K_3$ 과  $K_4$ 를 키 스케줄러로부터 입력받으면 시간  $[t_2-t_4]$ 구간에서 루우프 전개 사이퍼 함수를 연산하여 시간  $t_4$ 에서 연산된  $b_3$ 과  $b_4$ 값을 레지스터 A와 B에 저장할 수 있다.

<82> 이와 같이 시간  $t_0$ 로부터  $b_1$ 과  $b_2$  값을 계산하기 시작하여  $b_3, b_4, \dots, b_{16}$ 를 계산하여 해당 레지스터에 저장하여, 8 사이클 뒤인 시간  $t_{14}$ 에  $b_{15}$ 와  $b_{16}$ 을 레지스터 A와 B에 저장함으로써  $a_0$ 와  $b_0$ 를 입력으로 한 DES 연산을 끝낸다. 동시에 시간  $t_{16}$ 에서 다시  $c_0$ 와  $d_0$ 를 입력으로 한 DES 연산을 수행한다.

<83> 도14는 매크로 파이프라인의 효과를 나타낸 타이밍도이다.

<84> 상기 도14를 참조하면, 매크로 파이프라인을 사용할 경우에 8 라운드로 동작하는 이들 아키텍처와 일반적인 16라운드로 동작하는 DES 아키텍처의 성능을 비교 분석한 것이다. 한 개의 비암호화 입력 데이터를 입력하는 과정에서부터 DES 연산하여 출력하는



과정까지 걸리는 클럭 수를 레이턴시라고 하고 클럭 사이클 당 처리할 수 있는 비암호화 입력 데이터의 갯수를 처리능력비라고 하면 도14의 레이턴시와 처리능력비의 관계는 다음과 같다.

<85> 기존의 16 라운드 DES 아키텍처는 매크로 파이프라인을 사용하지 않을 경우, 입력 과정과 출력 과정에 각각 8 클럭 사이클이 걸리고, DES 연산과정에 16클럭 사이클이 걸리므로, 32클럭 사이클마다 새로운 비암호화 입력 데이터를 입력할 수 있다. 따라서, 이 경우에는 레이턴시는 32이고, 처리능력비는 1/32이다. 만약, 입력과 출력과정에 2단계 매크로 파이프라인을 사용하면, 레이턴시는 32로 상기의 경우와 동일하지만, DES 연산된 결과를 출력하는 과정과 새로운 비암호화 입력 데이터를 입력하는 과정을 동시에 수행할 수 있어서, 24 클럭 사이클마다 새로운 비암호화 입력 데이터를 입력할 수 있다. 따라서, 이 경우에는 레이턴시는 32이고 처리능력비는 1/24이다. 입력과정, DES 연산과정, 출력과정의 3단계 매크로 파이프라인을 적용하면, 입출력과정에서 각각 8 클럭 사이클씩 아이들(Idle)하게 되어 레이턴시는 40으로 늘어나지만, 처리능력비는 1/16으로 증가한다. 즉 16 사이클마다 새로운 비암호화 입력 데이터를 입력받아 DES 연산을 할 수 있다. 루우프 전개 사이퍼 함수를 이용한 DES 아키텍처와 시분할 사이퍼 함수를 이용한 DES 아키텍처는 입출력 과정과 DES 연산과정 모두 8클럭 사이클에 수행한다. 이들 이키텍처에 3 단계 매크로 파이프라인을 적용하면, 레이턴시는 24이고 처리능력비는 1/8로서, 8클럭 사이클마다 새로운 비암호화 입력 데이터를 암호화할 수 있다.

<86> 본 발명의 기술 사상은 상기 바람직한 실시예에 따라 구체적으로 기술되었으나 상기한 실시예는 그 설명을 위한 것이며 그 제한을 위한 것이 아님을 주의하여야 한다. 또

한, 본 발명의 기술 분야의 통상의 전문가라면 본 발명의 기술 사상의 범위내에서 다양한 실시예가 가능함을 이해할 수 있을 것이다.

**【발명의 효과】**

<87>       상기와 같이 본 발명은 기본 연산부 하나로 구성된 키 스케줄러를 사용함으로써 레지스터가 차지하는 면적을 반으로 줄여 집적화 시에 코스트(Cost)를 절감할 수 있도록 한다.

**【특허청구범위】****【청구항 1】**

데이터 암호화 표준 알고리즘을 이용하여 암호화 연산을 수행하는 암호화 장치에 있어서,

56 비트의 키를 입력받아 치환하는 제1치환선택부;

상기 제1치환선택부에 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭에 의해서 저장하는 제1레지스터 및 제2레지스터;

상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 각각 왼쪽으로 미리 정한 비트수만큼 쉬프트하는 제1쉬프트부 및 제2쉬프트부;

상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 입력받아 48비트의 보조키를 생성하는 제2치환선택부;

상기 제1 및 제2레지스터의 출력을 입력받아서 왼쪽으로 한자리 또는 두자리씩 쉬프트하는 제3 및 제4쉬프트부; 및

상기 제3 및 제4쉬프트부의 출력을 입력받아서 치환하는 상기과 별개의 제2치환선택부

를 포함하여 이루어진 키 스케줄러.

**【청구항 2】**

제 1 항에 있어서,

i번째 라운드에서 두개의 연속된 보조키를 얻기 위한 누적된 쉬프트 비트수는 1비트 또는 2비트의 차이가 있는 것을 이용한 것을 특징으로 하는 키 스케줄러.

【청구항 3】

제 1 항에 있어서,

상기 제2치환선택부는 와이어링으로 구현되는 것을 특징으로 하는 키 스케줄러.

【청구항 4】

제 1 항에 있어서,

상기 제2치환선택부는 보조키를 연산하는데 사용되는 레지스터에 저장된 값으로부터 다음 라운드의 보조키의 값을 연산하는 것을 특징으로 하는 키 스케줄러.

【청구항 5】

데이터 암호화 표준 알고리즘을 이용하여 암호화 연산을 수행하는 암호화 장치에 있어서,

56 비트의 키를 입력받는 제1치환선택부;

상기 제1치환선택부의 의해서 치환된 56비트의 블록을 28비트의 두 블록으로 나누어서 클럭에 의해서 저장하는 제1레지스터 및 제2레지스터;

상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 각각 왼쪽으로 미리 정한 비트수만큼 쉬프트하는 제1쉬프트부 및 제2쉬프트부;

상기 제1레지스터 및 제2레지스터의 28비트의 키 블록을 입력받아 48비트의 보조키를 생성하는 제2치환선택부;

상기 제1 및 제2레지스터의 출력을 입력받아서 오른쪽으로 한자리 또는 두자리씩 쉬프트하는 제3 및 제4쉬프트부; 및

상기 제3 및 제4쉬프트부의 출력을 입력받아서 치환하는 상기와 별개의 제2치환선택부

를 포함하여 이루어진 키 스케줄러.

#### 【청구항 6】

제 5 항에 있어서,

i번째 라운드에서 두개의 연속된 보조키를 얻기 위한 누적된 쉬프트 비트수는 1비트 또는 2비트의 차이가 있는 것을 이용한 것을 특징으로 하는 키 스케줄러.

#### 【청구항 7】

제 5 항에 있어서,

상기 제2치환선택부는 와이어링으로 구현되는 것을 특징으로 하는 키 스케줄러.

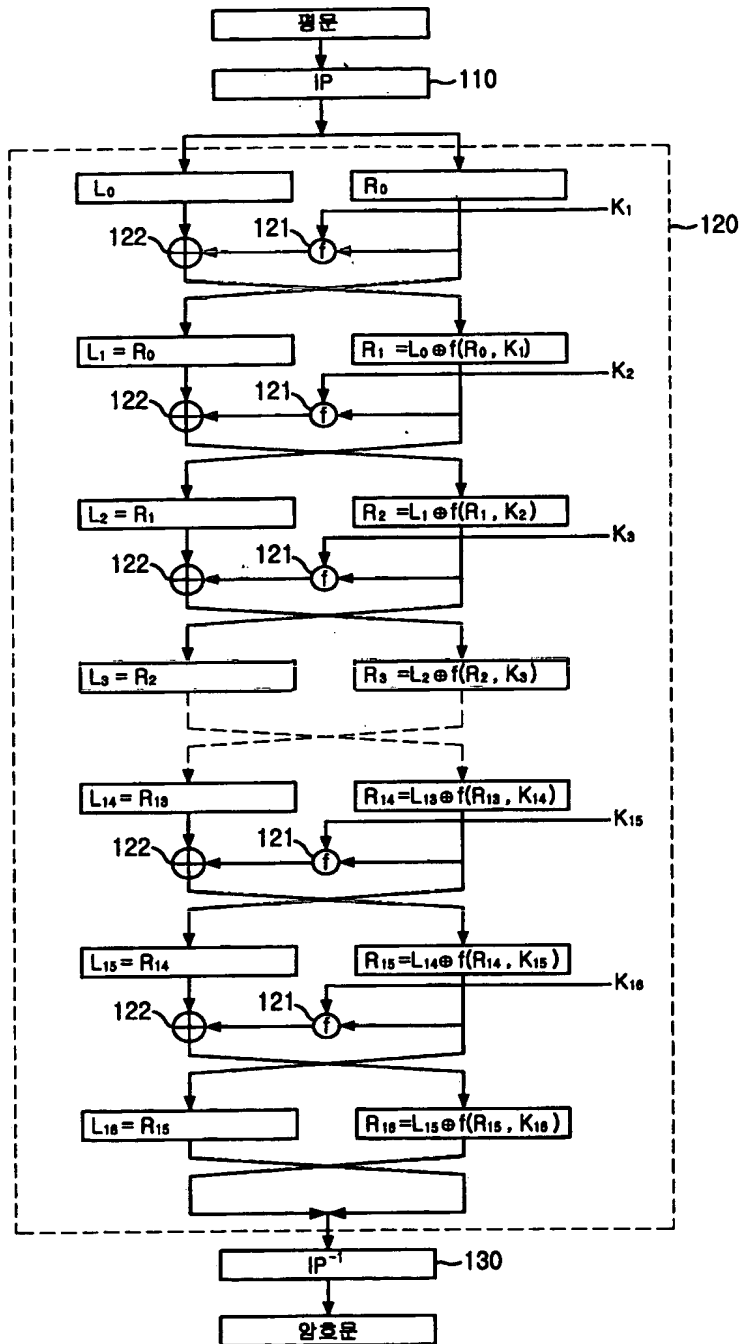
#### 【청구항 8】

제 5 항에 있어서,

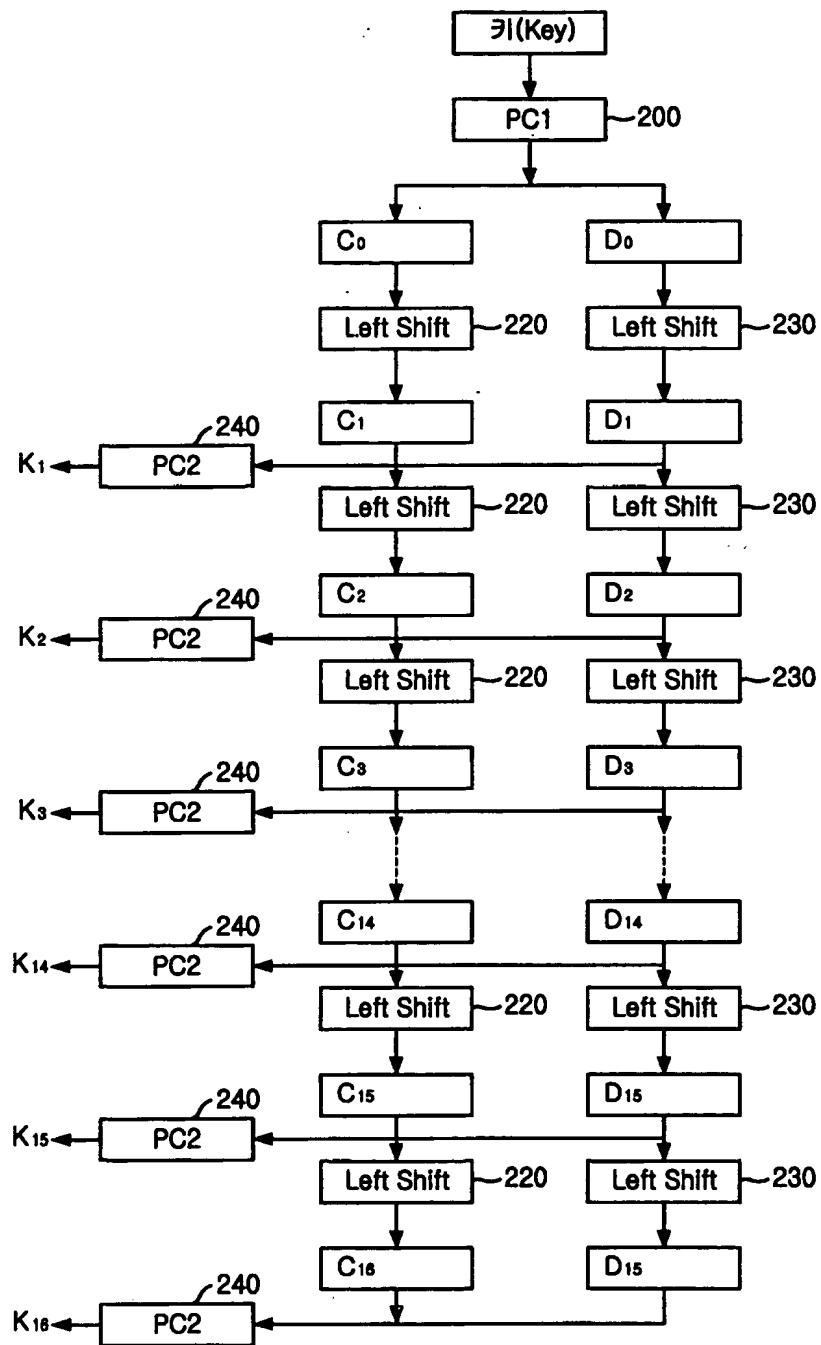
상기 제2치환선택부는 보조키를 연산하는데 사용되는 레지스터에 저장된 값으로부터 다음 라운드의 보조키의 값을 연산하는 것을 특징으로 하는 키 스케줄러.

## 【도면】

【도 1】

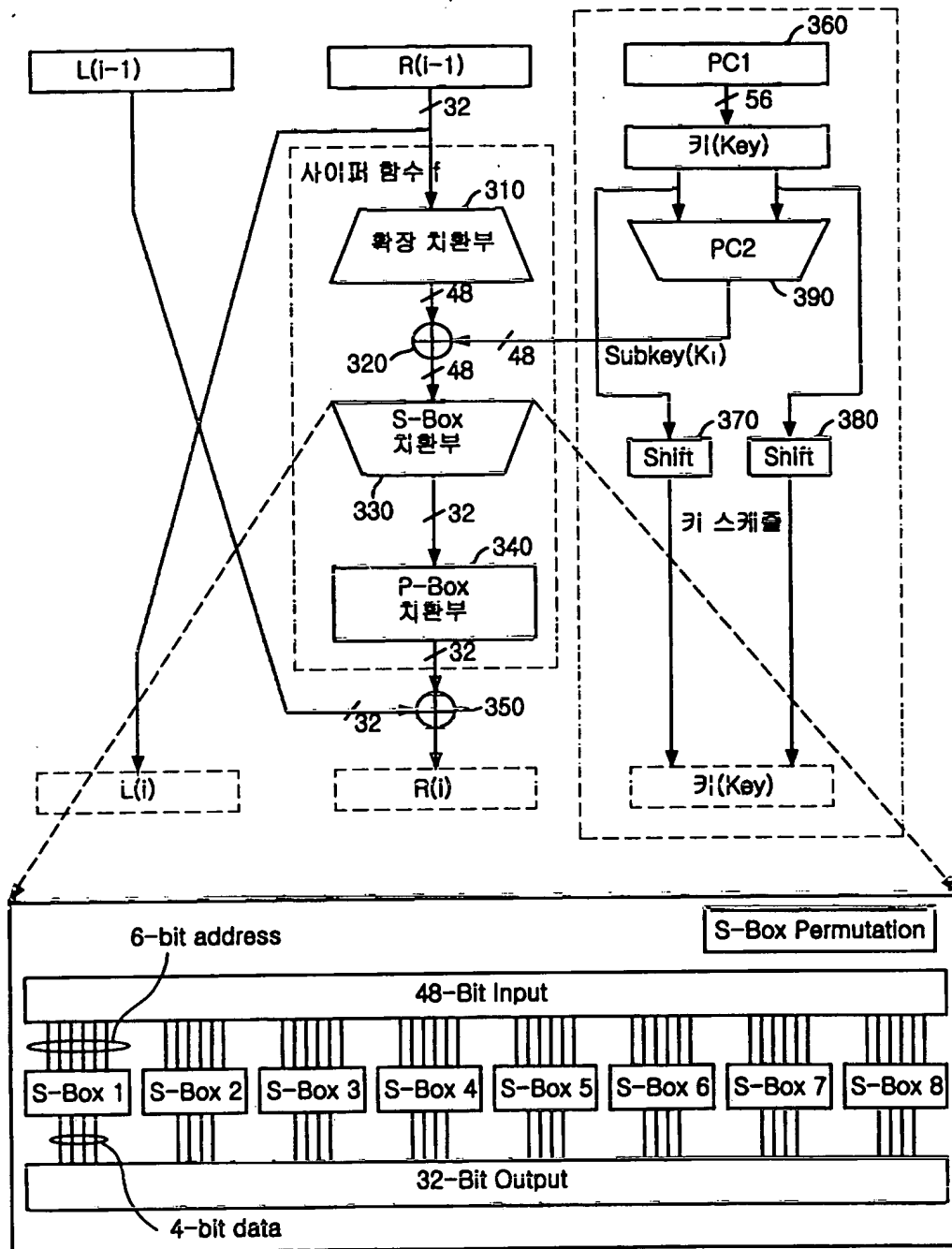


【도 2】

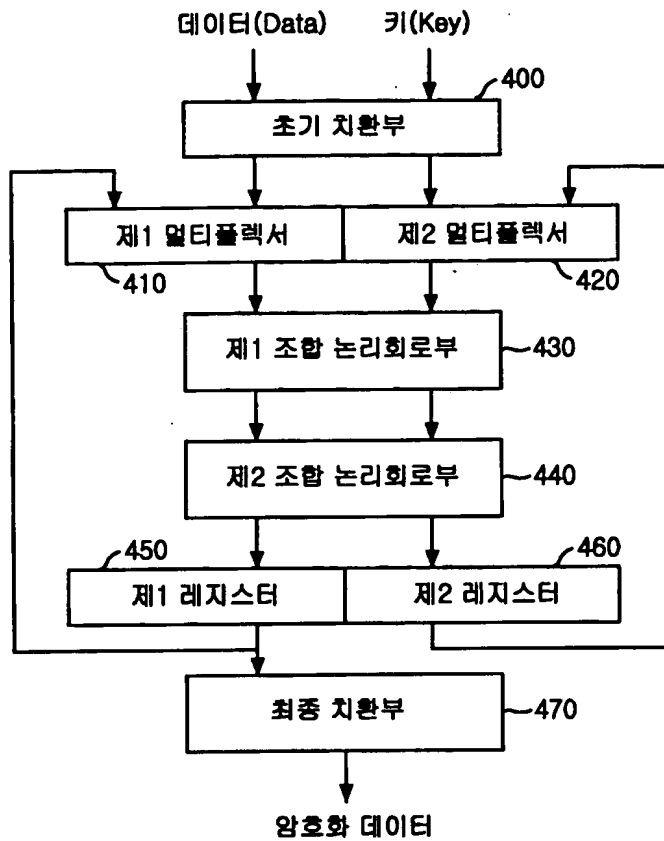




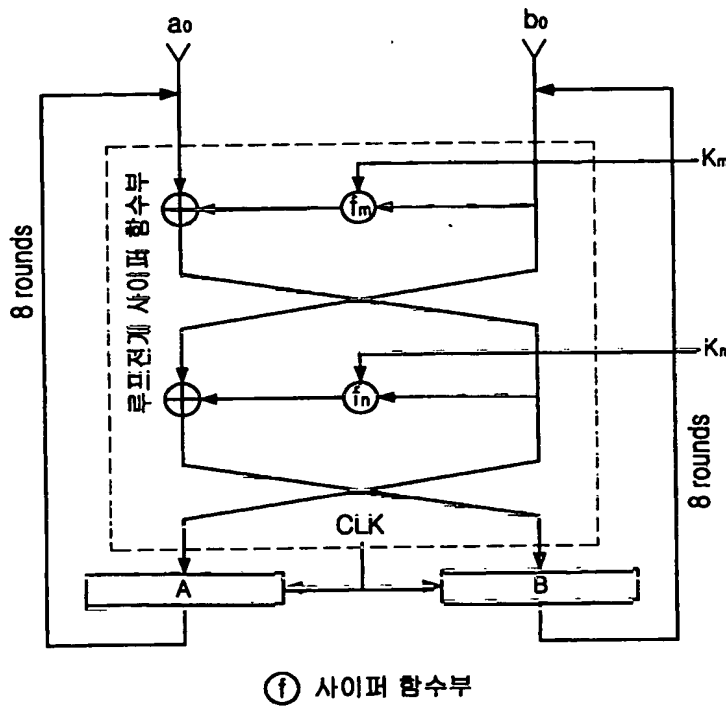
【도 3】



【도 4】

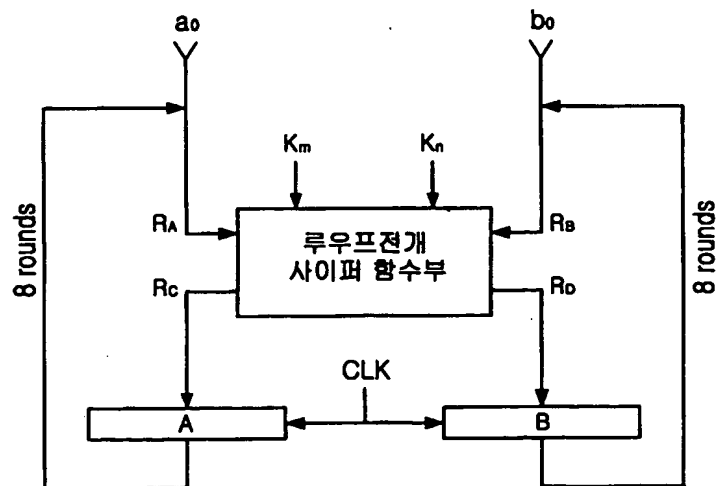


【도 5a】

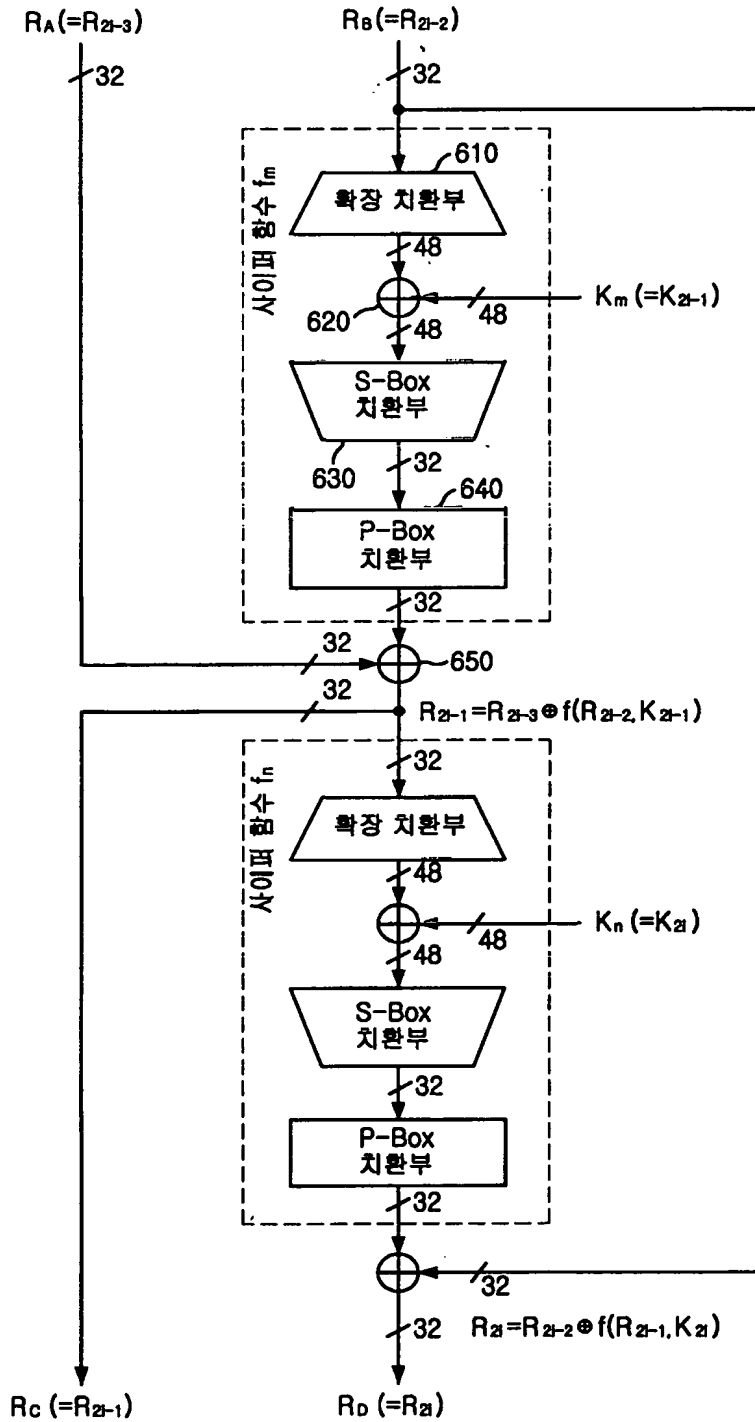


① 사이퍼 함수부

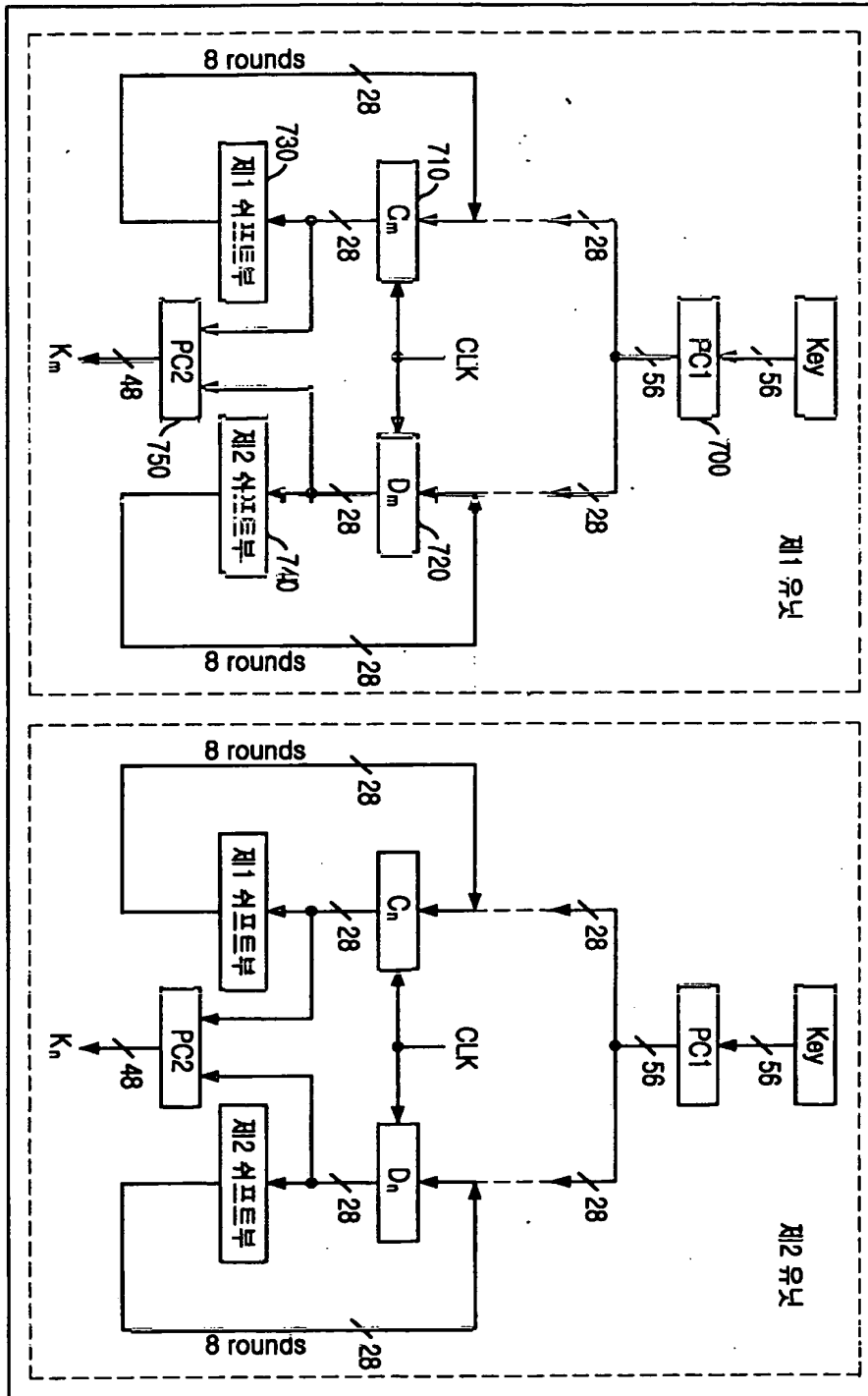
【도 5b】



【도 6】

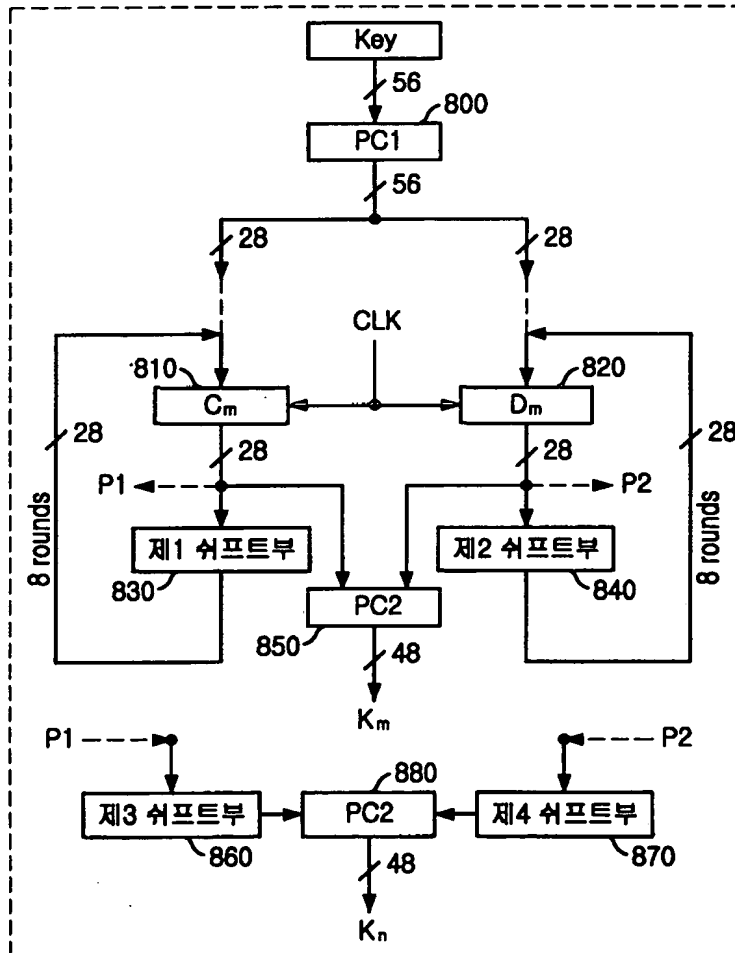


【도 7】



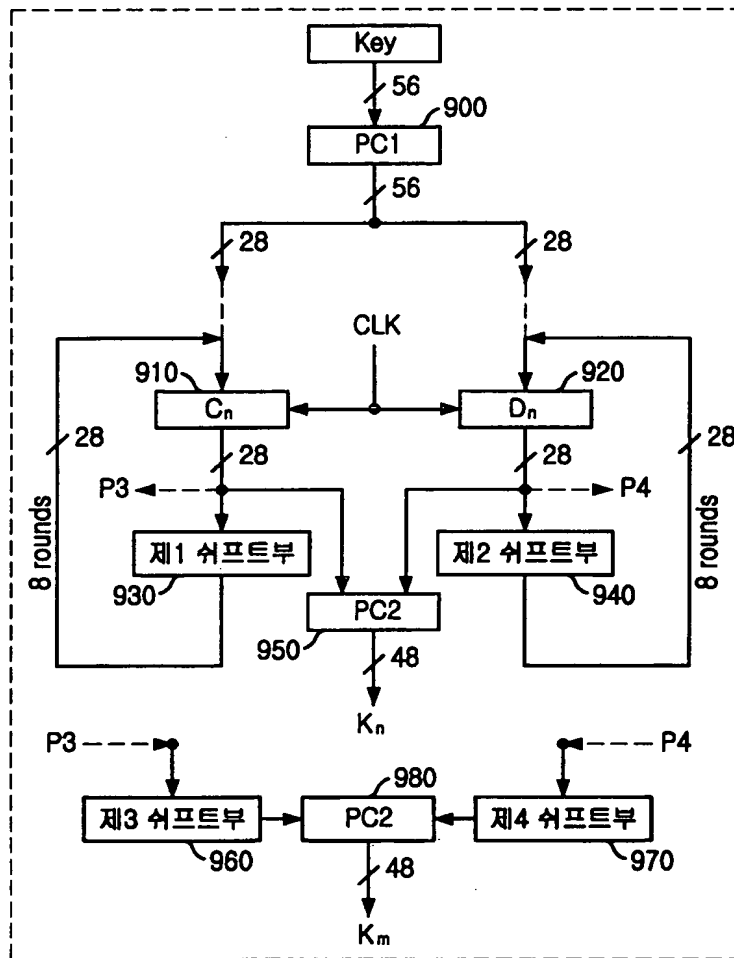
【도 8】

Round	$S_m$	$TS_m$	$D_m$	$TS_n$
1( $P_0$ )	3	1	+1	2
2( $P_1$ )	4	4	+2	6
3( $P_2$ )	4	8	+2	10
4( $P_3$ )	3	12	+2	14
5( $P_4$ )	4	15	+2	17
6( $P_5$ )	4	19	+2	21
7( $P_6$ )	4	23	+2	25
8( $P_7$ )	$2*(1)$	27	$+1(0)$	0

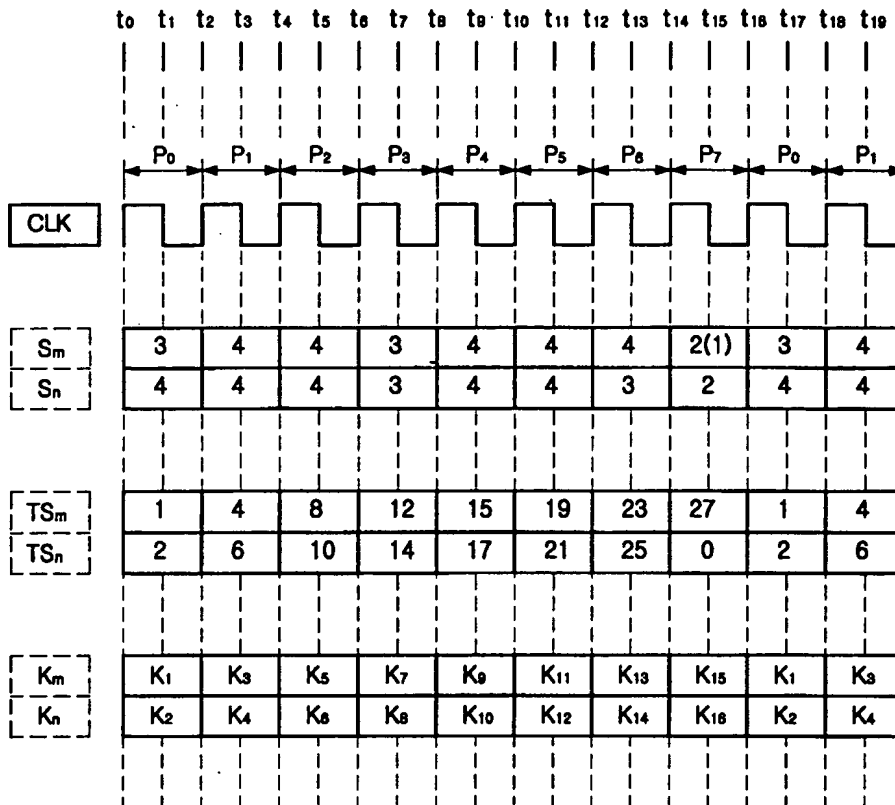


【도 9】

Round	$S_n$	$TS_n$	$D_n$	$TS_m$
1( $P_0$ )	4	2	-1	1
2( $P_1$ )	4	6	-2	4
3( $P_2$ )	4	10	-2	8
4( $P_3$ )	3	14	-2	12
5( $P_4$ )	4	17	-2	15
6( $P_5$ )	4	21	-2	19
7( $P_6$ )	3	25	-2	23
8( $P_7$ )	2	0	-1	27

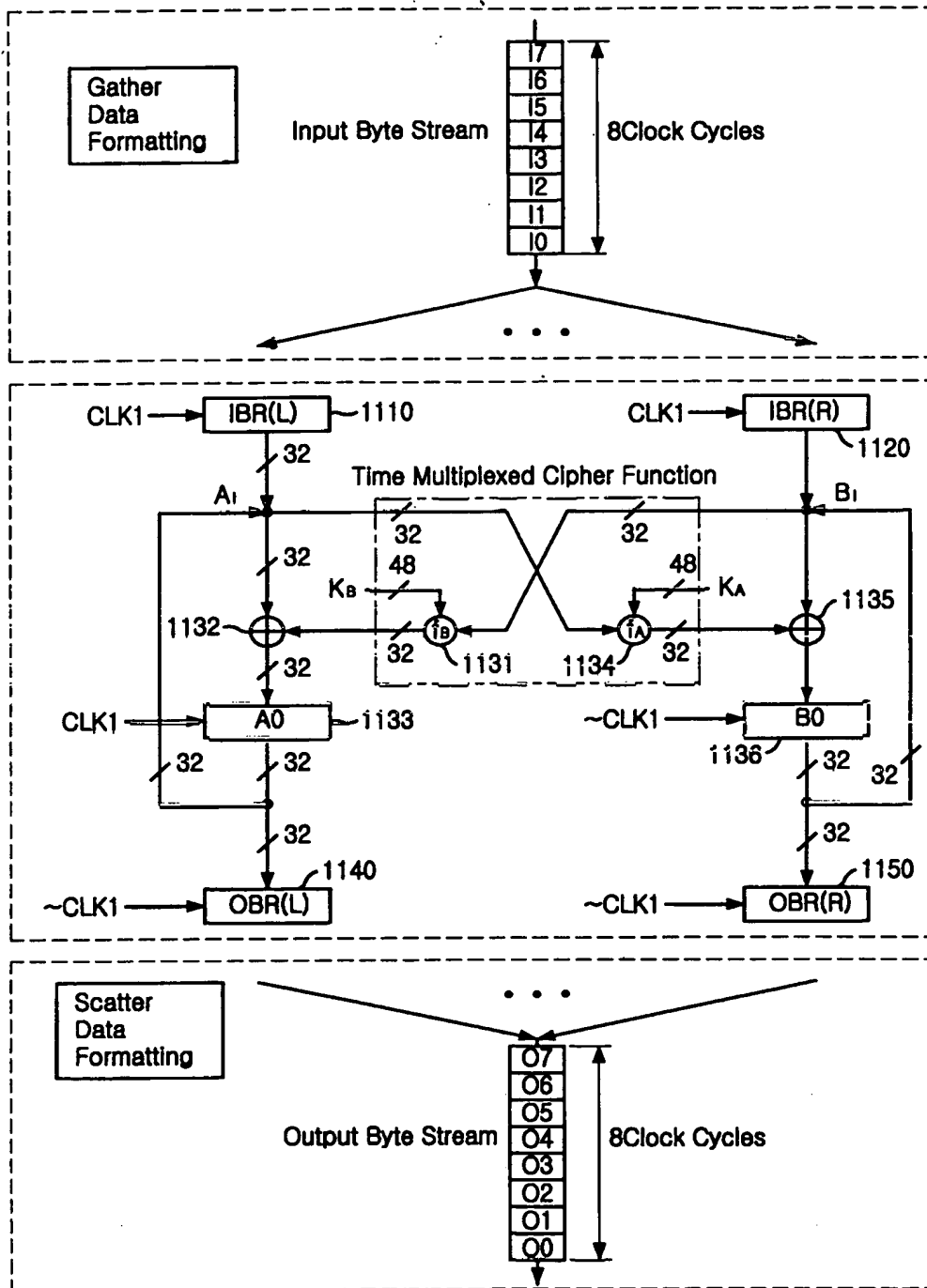


【도 10】

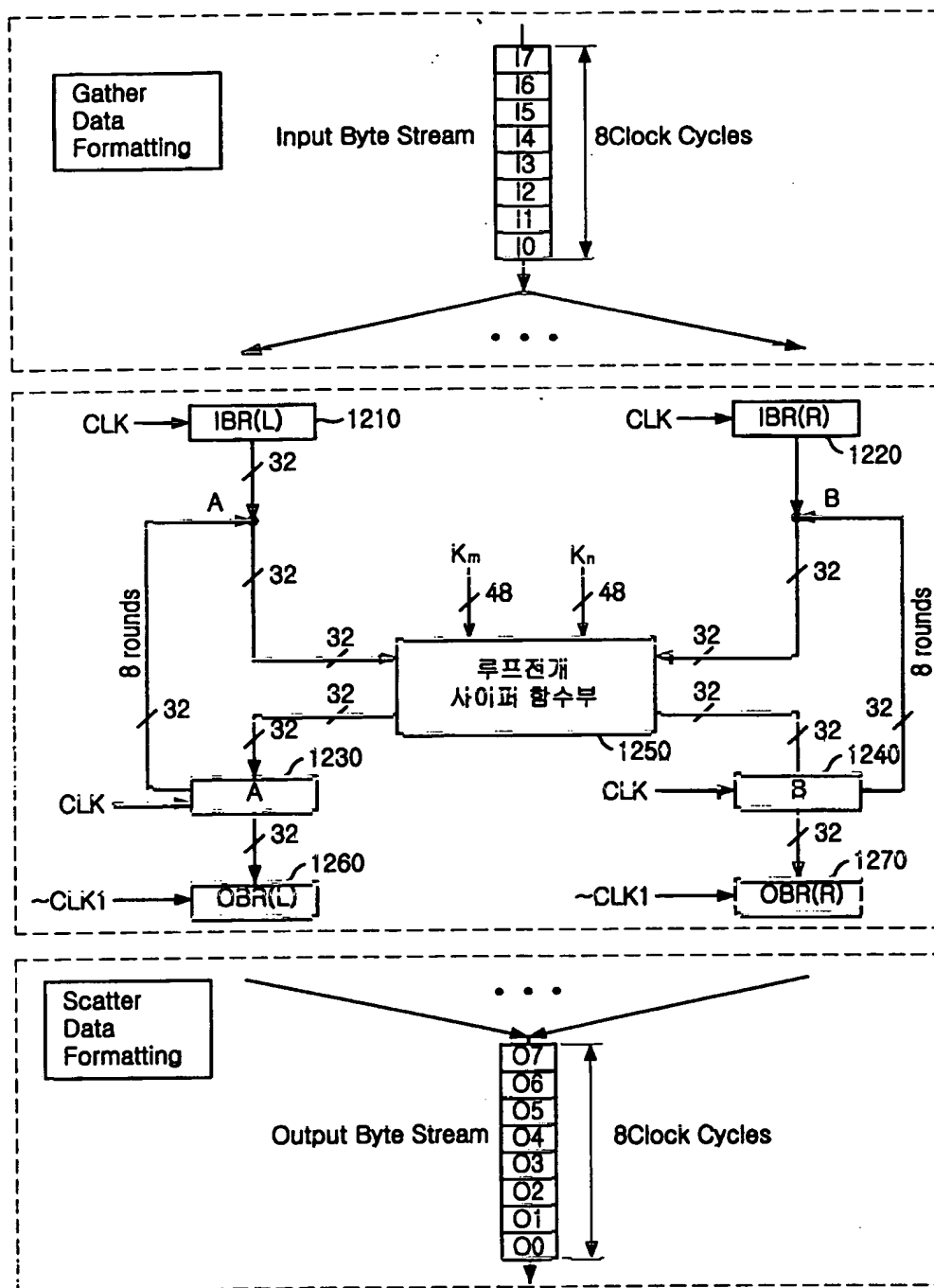




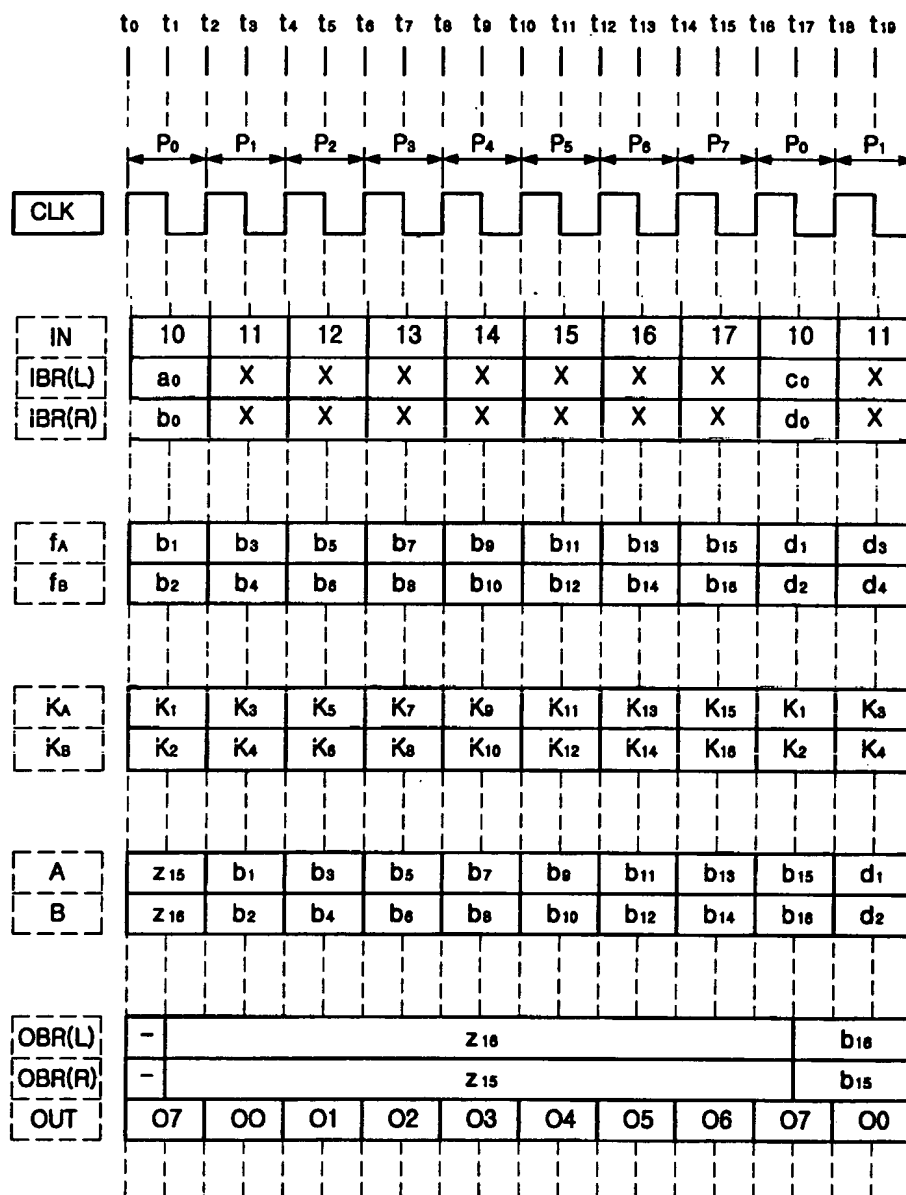
【도 11】



【도 12】



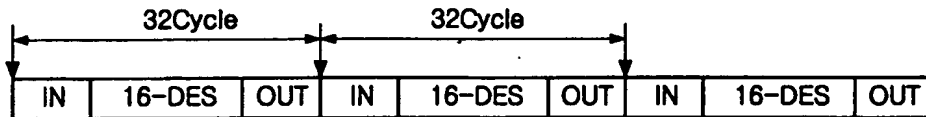
【도 13】



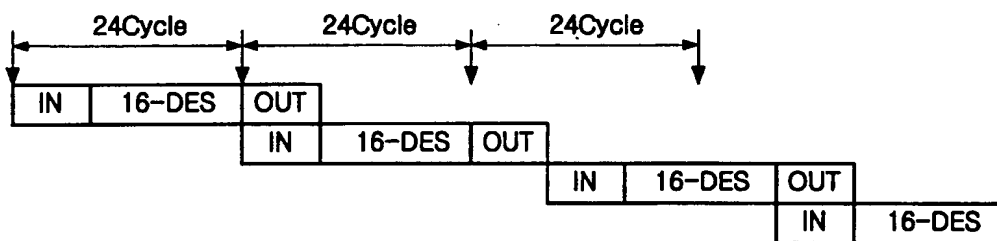
## 【도 14】

## Traditional 16 Round DES

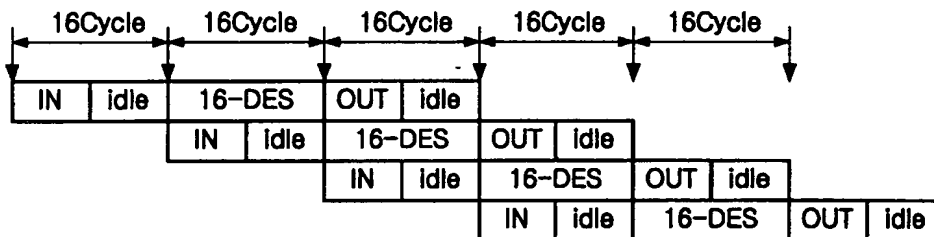
(1) No Latency Hiding (Latency=32, Throughput=1/32)



(2) 2-Stage Macro Pipeline (Latency=32, Throughput=1/24)



(3) 3-Stage Macro Pipeline (Latency=40, Throughput=1/16)



## 8Round DES (Loop Unrolled Cipher function &amp; Time Multiplexed Cipher Function)

(4) 3-Stage Macro Pipeline (Latency=24, Throughput=1/8)

